

Write up

For my project, I ran several test cases to ensure everything was working correctly. I tested adding courses, handling duplicate entries, reading courses from a file, and displaying all courses. For example, I tested adding "CMSC203" with CRN 30504 twice to see if the system prevents duplicates, and I also checked if the courses were displayed correctly in the GUI.

Initially, the JavaFX GUI showed null outputs, indicating issues with data processing. After debugging and refining my code, the GUI correctly displayed the course details. For instance, after adding "CMSC203" with CRN 30504 and "CMSC204" with CRN 30559, the GUI showed these courses accurately with all their details.














The expected output was a list of courses with correct details displayed in the GUI. Initially, the actual output didn't match due to issues like improper handling of duplicates and formatting errors. After making adjustments, the actual output matched the expected results, showing each course correctly and preventing duplicate entries.

I believe my project can pass a set of private test cases because I've handled common issues like invalid inputs and data integrity and provided clear error messages to users.

Through this project, I learned the importance of thorough testing and clear error handling. I realized how small issues can cause significant problems and how crucial it is to handle exceptions properly. I assumed users would input data in a specific format, which helped guide my error handling strategy.

Overall, this project has taught me the importance of detailed testing and error handling, which are crucial for reliable and user-friendly software.

GITHUB upload:

Project	File Name	Upload Date
 CourseDBElement.java	Add files via upload	now
 CourseDBGUI.java	Add files via upload	now
 CourseDBManager.java	Add files via upload	now
 CourseDBManagerInterface.java	Add files via upload	now
 CourseDBManagerTest.java	Add files via upload	now
 CourseDBManager_GFA_Test.java	Add files via upload	now
 CourseDBManager_STUDENT_Test.java	Add files via upload	now
 CourseDBStructure.java	Add files via upload	now
 CourseDBStructureInterface.java	Add files via upload	now
 CourseDBStructureTest.java	Add files via upload	now
 FXMainPane.java	Add files via upload	now
 Test1.txt	Add files via upload	now
 courses-F23.txt	Add files via upload	now

Example of creating database from file:

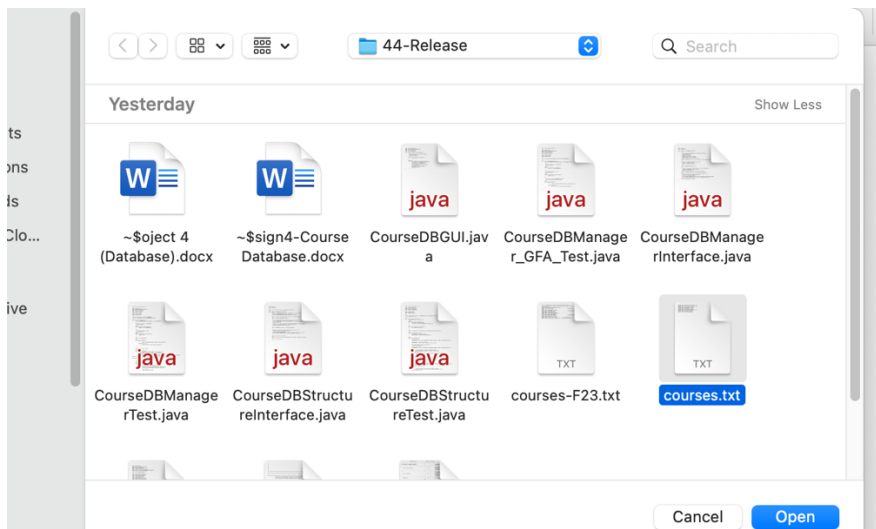
Course Database

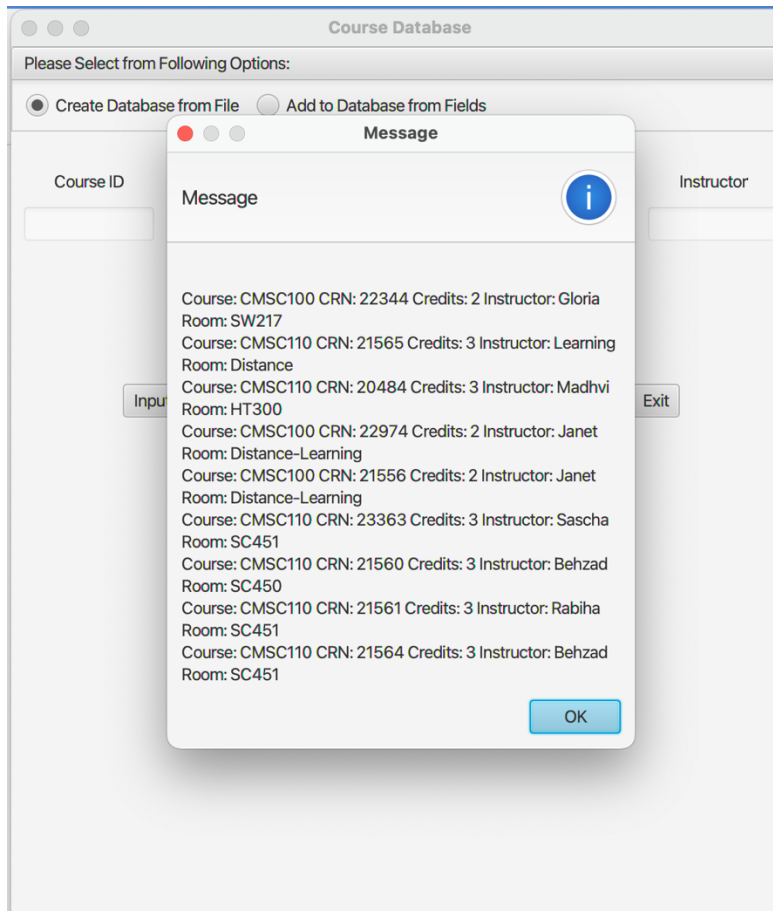
Please Select from Following Options:

☒ Create Database from File ☐ Add to Database from Fields

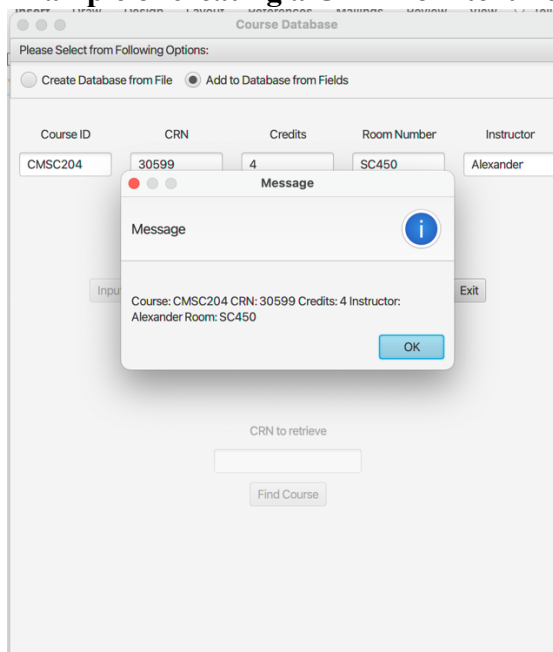
Course ID	CRN	Credits	Room Number	Instructor
<input type="text"/>	<input type="text"/>	<input type="text"/>	<input type="text"/>	<input type="text"/>

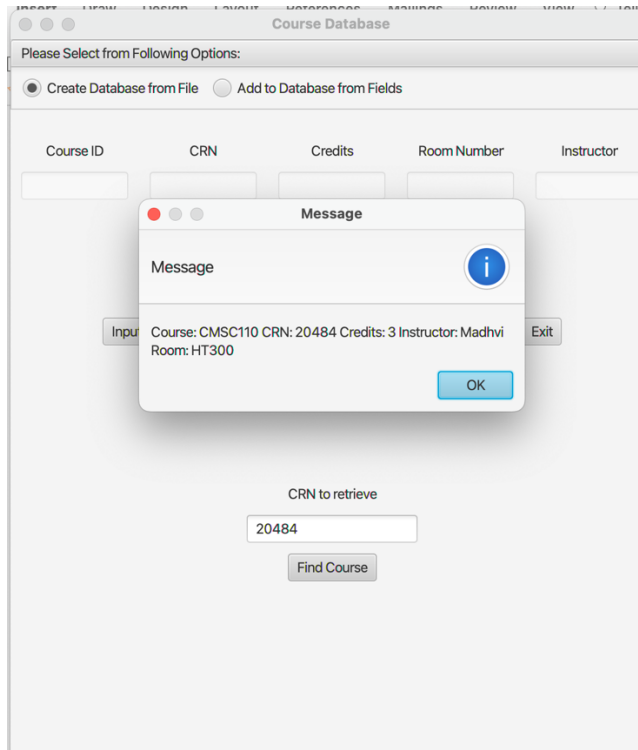
CRN to retrieve





Example of creating a CDE from text fields:

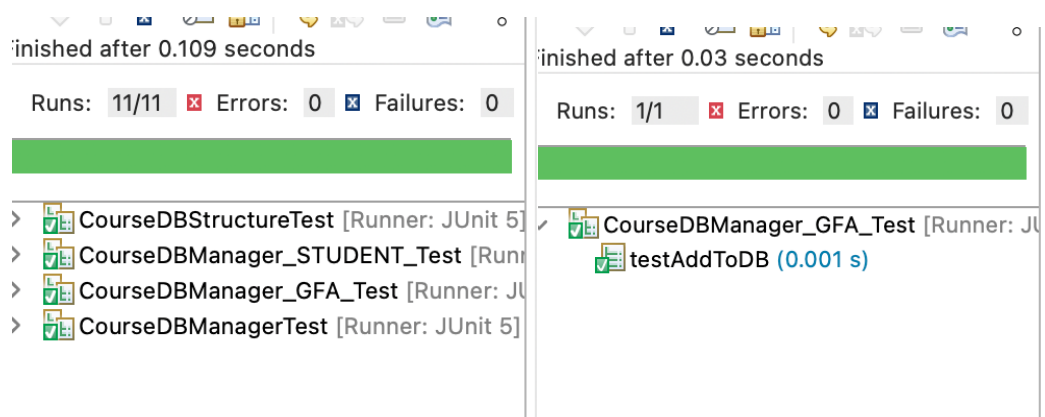




Course will not be uploaded to the database if (no error message will show but course will not be added):

- Input with credits outside the valid range (1 to 4)
- Input attempting to add a duplicate course
- Input with missing fields

JUNIT tests



finished after 0.055 seconds

Runs: 4/4 Errors: 0 Failures: 0

✓ CourseDBManager_STUDENT_Test [Runner: JUnit 5]

- ✓ testAddDuplicateCourse (0.001 s)
- ✓ testReadFile (0.000 s)
- ✓ testInvalidCourseLogging (0.000 s)
- ✓ testShowAll (0.000 s)

finished after 0.05 seconds

Runs: 3/3 Errors: 0 Failures: 0

✓ CourseDBManagerTest [Runner: JUnit 5]

- ✓ testAddToDB (0.001 s)
- ✓ testRead (0.000 s)
- ✓ testShowAll (0.000 s)

finished after 0.039 seconds

Runs: 3/3 Errors: 0 Failures: 0

✓ CourseDBStructureTest [Runner: JUnit 5]

- ✓ testGetNonExistent (0.000 s)
- ✓ testAddAndGet (0.000 s)
- ✓ testShowAll (0.000 s)